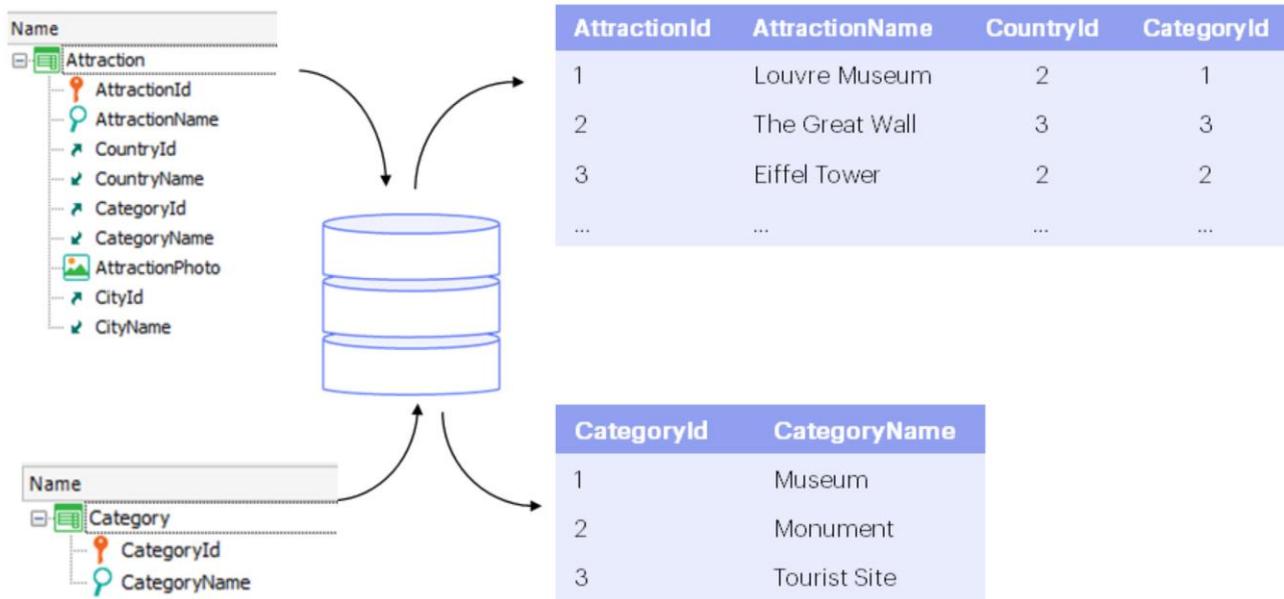


Popular de dados as tabelas a partir da própria  
transação

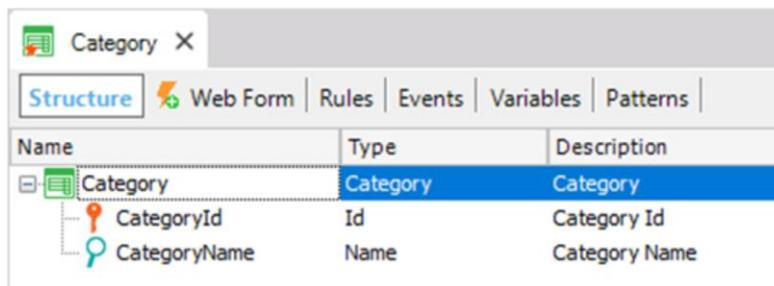
*GeneXus 16*

### Transações e Tabelas



Quando criamos uma transação, por padrão, GeneXus criará tabelas associadas para armazenar as informações que incluímos através de sua tela.

## GeneXus oferece uma solução para inicializar os dados de uma transação



Name	Type	Description
Category	Category	Category
CategoryId	Id	Category Id
CategoryName	Name	Category Name



Data	
Data Provider	False
Update Policy	Updatable

→

Data	
Data Provider	True
Used to	Populate data
Update Policy	Updatable

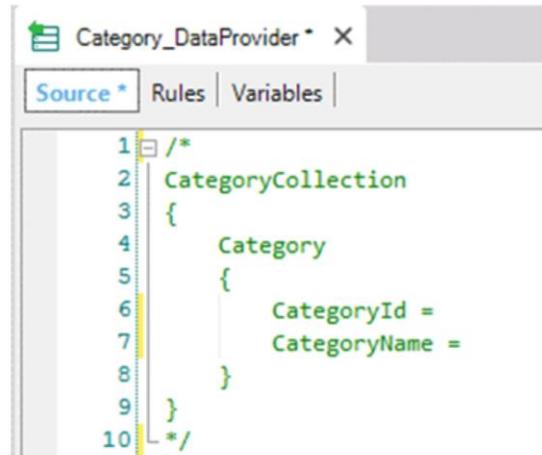
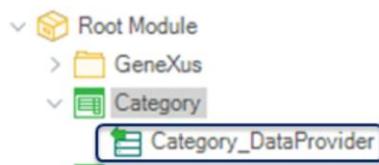
No vídeo anterior, vimos que estas tabelas poderiam ser inicializadas usando o Business Component associado à transação, por meio de uma variável coleção carregada usando um Data Provider.

Mas GeneXus já nos oferece uma solução para inicializar os dados correspondentes a uma transação, sem que tenhamos que fazer todas as etapas anteriores manualmente (obter o business component, criar o data provider, a variável coleção, chamar o data provider, fazer o Insert).

Para fazer isso, a transação conta com uma propriedade, abaixo do grupo Data, chamada **Data Provider**. Vamos ver com a transação Category. Por padrão está em False ... mas vamos passá-la para True.

Com isto, estamos dizendo que haverá um Data Provider associado. E nesta nova propriedade, informamos que o usaremos para inicializar os dados da tabela.

## Cria um Data Provider que usará o BC associado à transação



```
1  /*
2  CategoryCollection
3  {
4      Category
5      {
6          CategoryId =
7          CategoryName =
8      }
9  }
10 /*
```

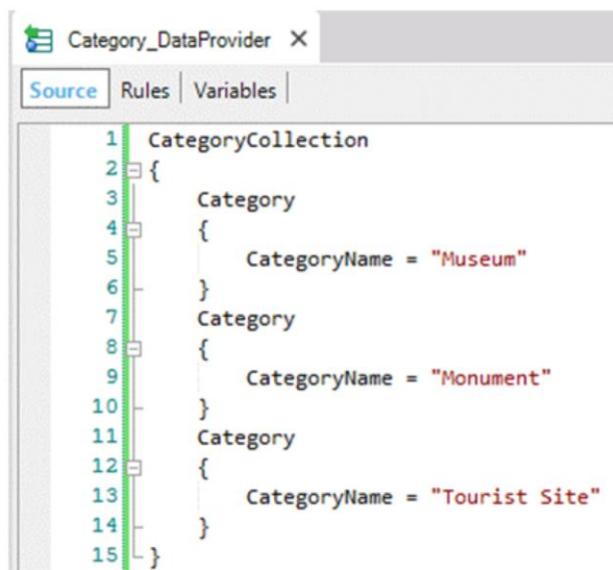
O código sugerido é quase o mesmo que o código que usamos no vídeo anterior...

Assim, vemos ao gravar que se criou um objeto do tipo Data Provider, que foi chamado Category\_DataProvider.

Além disso, se não tivéssemos a propriedade Business Component da transação em True, a teria colocado em True, fazendo que se crie, então, o business component associado com a transação.

Se abrirmos o Data Provider, podemos ver que já nos oferece o código para que simplesmente completemos os dados das categorias.

## Reutilizamos o código que usamos para completar o DP



```
1 CategoryCollection
2 {
3     Category
4     {
5         CategoryName = "Museum"
6     }
7     Category
8     {
9         CategoryName = "Monument"
10    }
11    Category
12    {
13        CategoryName = "Tourist Site"
14    }
15 }
```

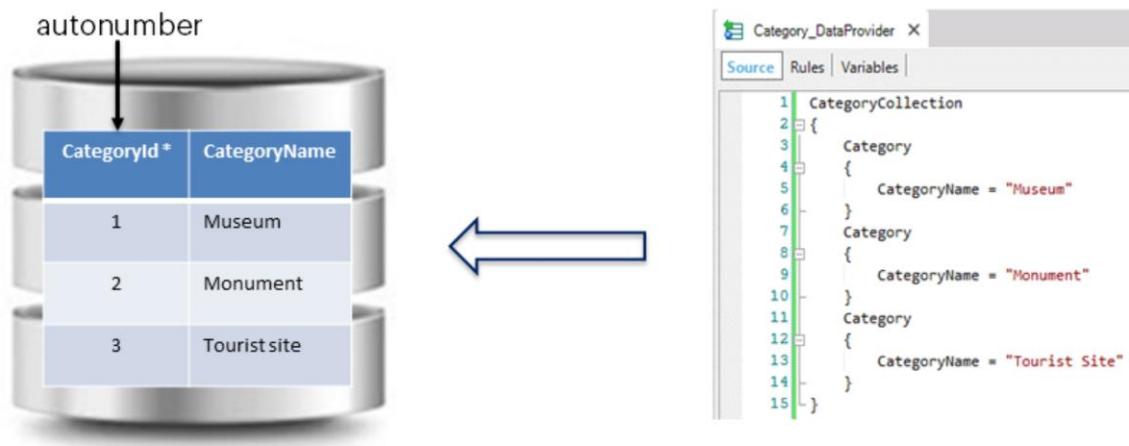
Este código é praticamente uma cópia do que fizemos antes manualmente.

Então copiamos o código que tínhamos escrito para o novo data provider.

Com isto, já definimos como o Data Provider irá atribuir valores para as novas categorias a serem criadas. O que ainda não razonamos é o momento em que esse Data Provider será chamado para executar a tarefa. Se pensarmos sobre isso, o momento certo será quando se cria a tabela no banco de dados.

No entanto, para não sobrecarregar o processo de criação das tabelas, o GeneXus adiará a execução do Data Provider até o momento em que se execute a aplicação, que será o momento em que realmente precisamos que os dados estejam na tabela.

GeneXus executa o Data Provider quando a tabela é criada no banco de dados.

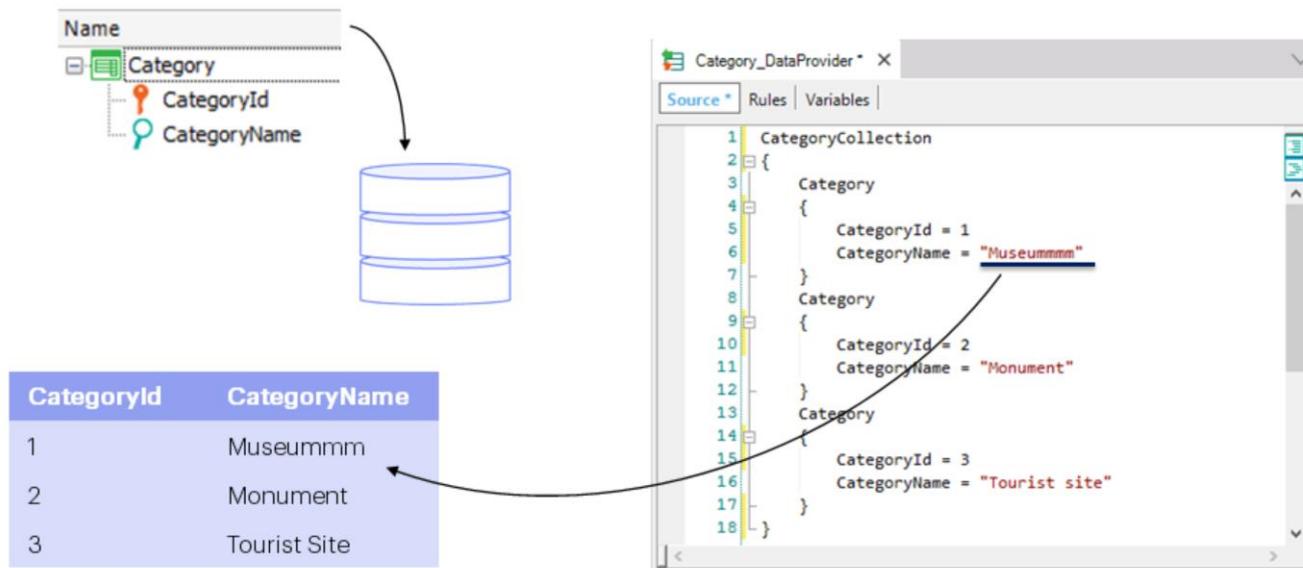


Mas no nosso caso a tabela já está criada e não terá dados porque vamos eliminá-los. Pressionamos Remove Data e com isso se eliminam os dados de Category e de Attraction.

No entanto, como acabamos de habilitar a inicialização dos dados da transação, em seguida, na próxima execução GeneXus executar irá executar o Data Provider.

Observemos que se a tabela já tiver dados nesse momento, porque temos o identificador como autonumber ... o que fará será adicionar novos registros. Ou seja, não importa se já existia uma categoria Museum, irá inserir outra. Não é nosso caso, já que tivemos a precaução de esvaziar a tabela antes. Caso contrário, se o identificador não fosse autonumber teríamos que indicar seu valor para cada novo grupo do data provider, e se já existissem na tabela registros com os valores que estamos adicionando, serão atualizados.

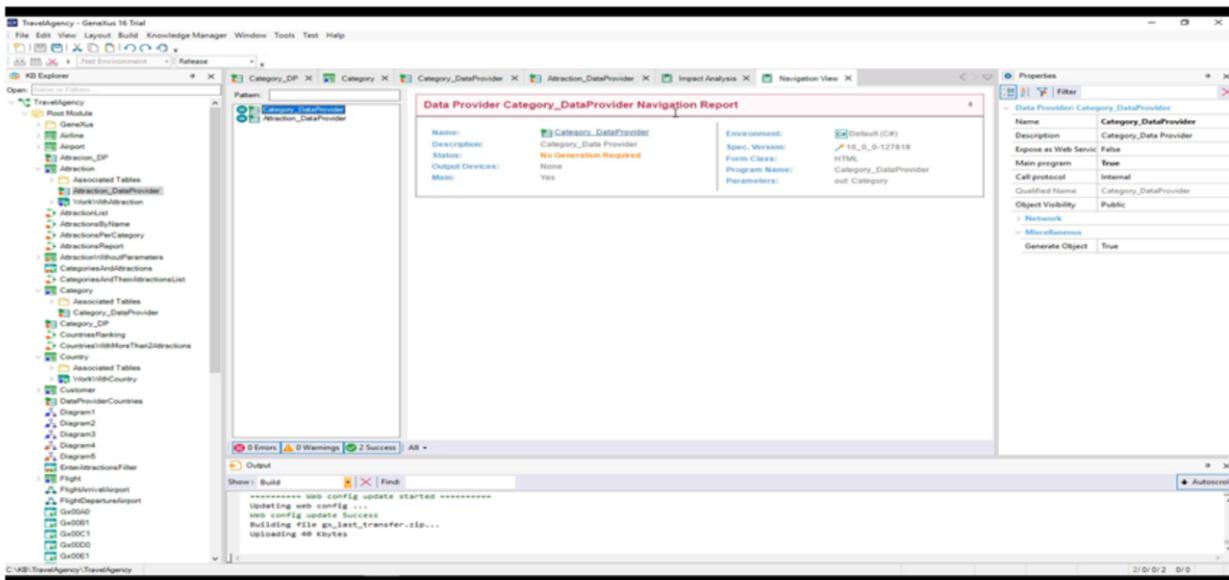
### Preenchendo Tabelas



É que ao tentar fazer a inserção via Business Component, encontrará chave duplicada e o que faz é um update. Por isso, neste exemplo, mudaria o valor do nome da categoria "Museum" por este outro com muitos emes.

Bem, para preencher com dados a transação Attraction faremos o mesmo que fizemos antes com Category.

DEMO



[ DEMO: <https://youtu.be/MhzFWrA7Ulw> ]

Vamos executar o que foi feito até agora. F5.

Vemos que a lista de navegação informa que deverá ser executado o programa de inicialização da tabela Category. E também da tabela Attraction.

E vemos na execução que efetivamente executou esses programas e voltamos a ter dados nas tabelas (observemos os identificadores! Lembremos que são autonumerados)

Bem, se agora ou mais tarde precisamos modificar o Data Provider de inicialização, adicionando, por exemplo, uma categoria que não tínhamos contemplado inicialmente ... ao fazer F5 GeneXus notará que alterou o data provider e voltará a executá-lo.

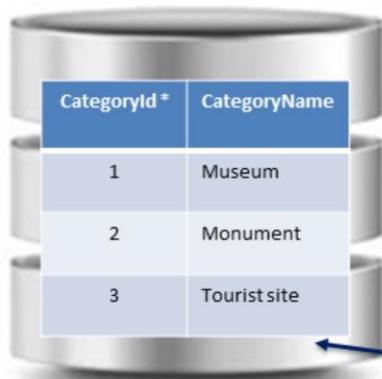
Mas, ao fazê-lo, como já existiam as categorias Museum, Monument e Tourist site na tabela, com ids autonumerados, então voltará a inseri-las com novos ids, além de inserir a nova. Temos uma maneira de evitar isso, criando um índice Unique para o nome da categoria, para que se verifique antes de inserir o registro que esse valor não esteja repetido. Não faremos isso aqui.

Executemos e excluimos manualmente os novos registros duplicados.

Observemos que a transação segue funcionando, em tudo o mais, de forma padrão. Ou seja, continuaremos inserindo, atualizando e excluindo seus dados através da tela, como sempre

Suas regras serão executadas, e será capaz de usar o business component associado, como fizemos antes. O que vimos só afeta a inicialização de seus dados.

Se o DP é trocado, GeneXus toma conhecimento e volta a executá-lo



CategoryId *	CategoryName
1	Museum
2	Monument
3	Tourist site

```
CategoryCollection
{
  Category
  {
    CategoryName = "Museum"
  }
  Category
  {
    CategoryName = "Monument"
  }
  Category
  {
    CategoryName = "Tourist site"
  }
  Category
  {
    CategoryName = "Nature reserve"
  }
}
```



## É possível que os dados de inicialização permaneçam inalterados

Para fazer isso, usamos a propriedade Update Policy definida como Read Only:

### ▼ Data

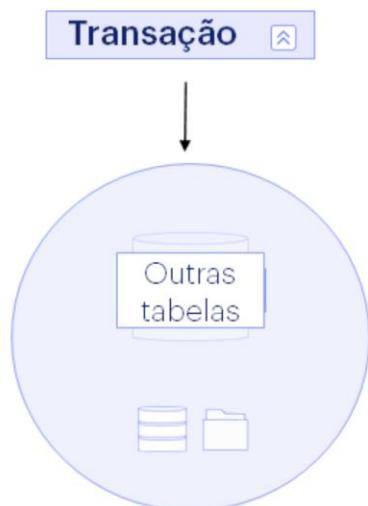
Data Provider	True
Used to	Populate data
Update Policy	Read Only

 United States
 Brazil
 Mexico
 Colombia
 Argentina
 Canada
 Peru
 Venezuela
 Chile
 Ecuador
 Guatemala
 Cuba
 Haiti
 Bolivia
 Dominican Republic
 Honduras
 Paraguay
 Nicaragua
 El Salvador
 Costa Rica
 Panama
 Puerto Rico
 Uruguay
 Jamaica
 Trinidad and Tobago

Mas se a transação corresponde a informações que não mudam ao longo do tempo, como por exemplo, os países, estados ou regiões de um país, parâmetros de um sistema, etc, é desnecessário que a transação ou o business component nos permita atualizar seus dados. Para garantir que os dados não sejam alterados, configuramos a propriedade "Update Policy" em Read Only.

## Transação Dinâmica

Se os dados da transação **são** obtidos de outras fontes, não haverá uma tabela associada.



### Usos da transação:

1. Insert, Update, Delete data
2. Navigate (retrieve) data

#### ▼ Data

Data Provider	True
Used to	Retrieve data
Update Policy	Read Only

Até aqui o uso da transação corresponde mais ou menos ao conhecido, onde a transação tem sua "tabela" associada.

Mas temos outros casos nos quais os dados da transação são obtidos de outras fontes, que podem ser consultas complexas ao banco de dados que inclui buscar informações em várias tabelas, ou consultas a outros bancos de dados, etc.

Neste caso, a transação não terá essa tabela associada. O Data Provider é quem se encarrega de obter esses dados. Para especificar este uso, a propriedade "Used to" configuramos com o valor "Retrieve data".

Essas transações são chamadas de "transações dinâmicas", sobre as quais não nos aprofundaremos neste curso.

Em conclusão, para popular de dados uma tabela, não o faríamos manualmente como fizemos no vídeo anterior, mas usando a propriedade Data Provider da transação.

Para finalizar, fazemos um Commit de nossas modificações no GeneXus Server.

# GeneXus™

**The power of doing.**

Videos

[training.genexus.com](http://training.genexus.com)

Documentation

[wiki.genexus.com](http://wiki.genexus.com)

Certifications

[training.genexus.com/certifications](http://training.genexus.com/certifications)